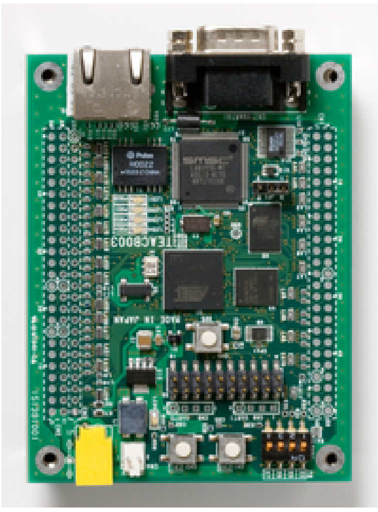
はじめてみよう μTeaboard/ARM7-AT91



Version 1.01

パーソナルメディア株式会社

目 次

修正履歴

Version 1.01

- 開発環境の更新に伴い、「2. 開発環境のインストール」を修正
- ネットワーク送受信サンプルを追加

Version 1.00

- μTeaboard/ARM7-AT91 向けに新規作成

はじめに

本書では、はじめて「μTeaboard（マイクロティーボード）」をご利用になる方を対象に、μTeaboard/ARM7-AT91 を使い始めるためのチュートリアルをご提供します。インストールから実習用プログラムの実行まで、この手順に沿って試していただければ、短時間で開発の手順をひととおり体験できます。

1 ターゲット側の動作確認

組込みシステム開発では、プログラムをメイクする開発環境側と、作成したプログラムを実行するターゲット側が、別々のマシンに分かれているのが一般的です。このような開発手法を「クロス開発」と呼びます。μTeaboard/ARM7-AT91 においても、μTeaboard 用ソフトウェアの開発をクロス開発で行います。(図 1.1)

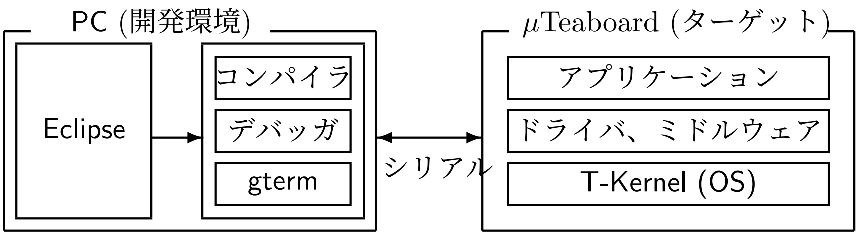


図 1.1 クロス開発

ターゲット側実行環境は、μTeaboard の出荷時にフラッシュ ROM に格納されています。まずこの章では、ターゲット側の動作確認を行います。

(1) シリアル接続

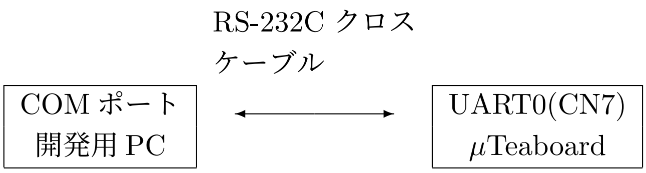


図 1.2 シリアル接続

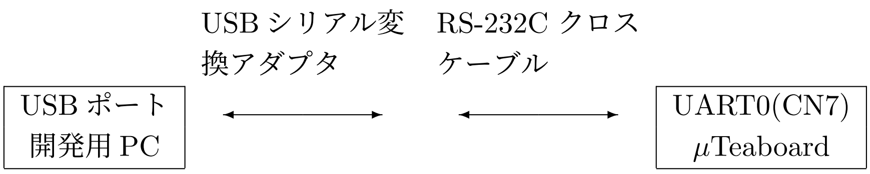


図 1.3 USB シリアル変換による接続

開発用パソコンと μTeaboard を、RS-232C クロスケーブルを使って図 1.2 のように接続します。パソコンにシリアルポート (COM ポート) がない場合は、USB シリアル変換アダプタを使用して図 1.3 のように接続してください。

(2) 端末ソフトの起動

パソコン上には端末ソフト (「ハイパーターミナル」や「Tera Term Pro」など) が必要です。

† Windows 8, 7 および Vista にはハイパーターミナルが付属しません。端末ソフトをお持ちでない場合は「Tera Term Pro (バージョン 2.3)」を下記からダウンロードしてご利用ください。

`http://hp.vector.co.jp/authors/VA002416/`

† この章では動作確認のためにハイパーターミナルなどの端末ソフトを使いますが、2 章で Eclipse の開発環境をインストールすれば、端末ソフトを使うかわりに Eclipse 上の `gterm` を使ってコンソール操作ができます。ただし、最初のセットアップ時には、接続がうまくいかない場合の原因切り分けのためにも、まずはハイパーターミナルなどの端末ソフトで接続を確認することをお勧めします。

- **Tera Term Pro の場合**

  Tera Term Pro を起動します。

- **ハイパーターミナルの場合**

  Windows XP 上のハイパーターミナルの場合は、「スタート」→「すべてのプログラム」→「アクセサリ」→「通信」→「ハイパーターミナル」を起動します。

  † 「既定の Telnet プログラムにしますか?」と聞かれる場合がありますが、「はい」でも「いいえ」でもどちらでも構いません。
  また市外局番などを聞かれる場合もありますが、これも実際に電話をかけるわけではありませんので、適当に入力して構いません。

  名前とアイコンの選択では、適当な名前を入力して OK をクリックします (図 1.4)。



図 1.4　ハイパーターミナルの起動

(3) 通信条件の設定

開発用パソコンで使用するシリアルポート (「COM1」など) を指定して、ポートの通信条件の設定を次のように設定します。

ビット/秒	38400 bps
データビット	8 ビット
パリティ	なし
ストップビット	1 ビット
フロー制御	XON/XOFF

- **Tera Term Pro の場合**

  「ファイル (F)」→「新規接続 (N)」を選択します。新規接続ダイアログが表示されますので、「シリアル」を選択し、使用するシリアルポート (「COM1」など) を指定して、 OK をクリックします。

  続いて「設定 (S)」→「シリアルポート (P)」を選択します。シリアルポート設定ダイアログが表示されますので、上記のように設定してください。

- **ハイパーターミナルの場合**

  接続方法に使用するシリアルポート (「COM1」など) を選択して OK をクリックします。続いてポートの設定ダイアログが表示されますので、図 1.5 のように設定してください。

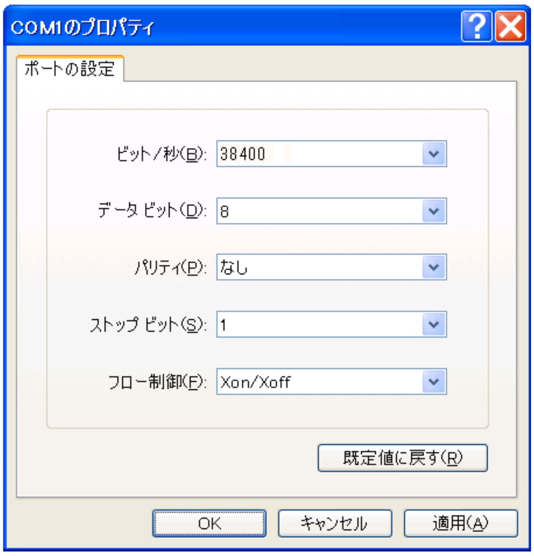


図 1.5 ハイパーターミナルの通信設定

(4) μTeaboard の起動

μTeaboard のディップスイッチ (DSW1) は 4 つとも OFF にします。
μTeaboard に 5V の AC アダプタを接続して電源を供給します。
μTeaboard の LED5 (緑) および LED1 (赤) が点灯します。

(5) シリアル接続の確認

端末ソフト上で ↩ (Enter) キーを何回か押してみて、IMS のプロンプト ([IMS]%) が表示されれば成功です (図 1.6)。

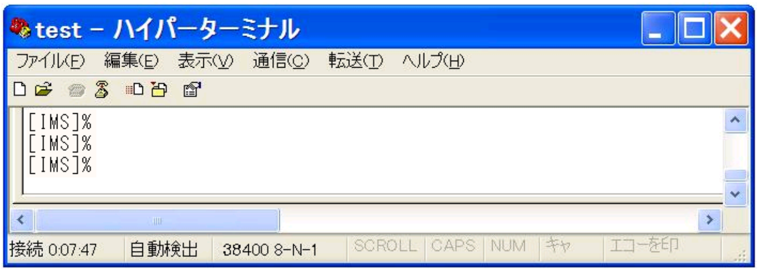


図 1.6　端末ソフト上に表示された IMS のプロンプト

2 開発環境のインストール

2.1 Cygwin のインストール

Windows パソコン上で開発環境を使うには、まず Cygwin をインストールする必要があります。

詳細については μTeaboard/ARM7-AT91 の CD-ROM 内の『Cygwin インストール方法説明書』をあわせてご参照ください。

† Windows のログイン時のアカウントは、管理者権限としてください。制限されたアカウントの場合、ソフトウェアのインストールがうまくいきません。

また Windows のログイン時のアカウント名は、半角英数字のみとしてください。アカウント名に全角文字や半角空白などが入ると、Cygwin で不都合が生じる場合があります。

† 既に Cygwin がインストールされている場合でも、バージョンが古かったり、必要なモジュールが足りない場合も考えられますので、『Cygwin インストール方法説明書』をご一読の上、必要に応じて再インストールをお願いいたします。

(1) パッケージの展開

μTeaboard/ARM7-AT91 の CD-ROM 内の共通ソフトウェアのフォルダ (`common¥soft`) にある Cygwin システムパッケージ「`cygwin_d-1.7.17-1.zip`」を、いったんデスクトップなどに保存した上で、Windows のエクスプローラでマウスの右ボタンをクリックして「すべて展開 (A)...」を実行してください。展開先はハードディスクのどこでも構いません。

† ファイル名の中のバージョン番号は上記と異なる場合があります。

(2) インストーラの起動

展開したフォルダ内にあるインストーラ `setup.exe` を起動します (図 2.1)。

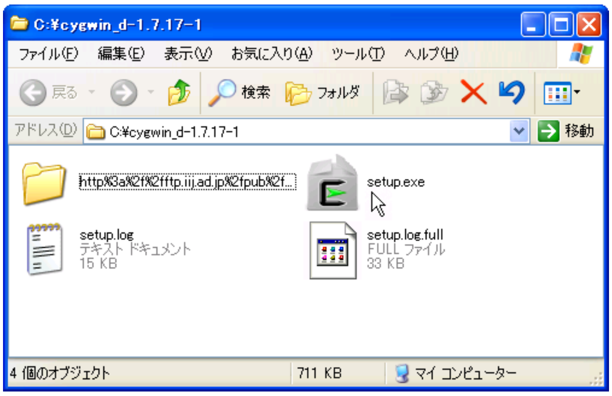


図 2.1 Cygwin インストーラの起動

インストーラが起動したら **次へ** をクリックします。

(3) インストールタイプの選択

「Install from Local Directory」を選択して **次へ** をクリックします。

(4) インストールディレクトリの選択

Cygwin をインストールするディレクトリを選択します。デフォルトでは「`C:¥cygwin`」になっています。支障がなければデフォルトのままにしておきます (図 2.2)。

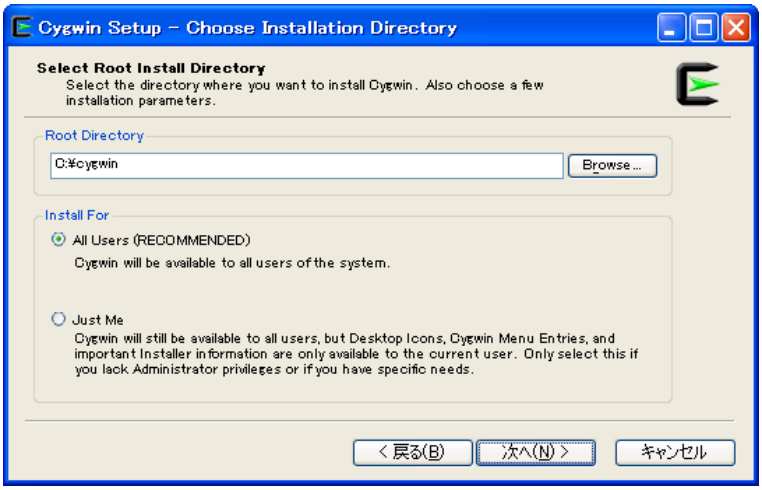


図 2.2 インストールディレクトリの選択

「Install For」は「All Users」を選択してください。これはデフォルトです。

次へ をクリックします。

(5) ローカルパッケージディレクトリの選択

(1) でインストールパッケージを展開したフォルダを指定して、**次へ** をクリックします。

(6) インストールするパッケージの選択

ファイルのチェックが行われた後、インストールパッケージの選択画面になります。

† Cygwin 1.7 系列の最初のインストール時には「Setup Alert」ダイアログが表示されます。内容をよく読んでから作業を行ってください。旧バージョンの Cygwin を使用している場合、何らかの影響が出る可能性があります。

「All」の右の「Default」をクリックして「Install」の状態にしてから **次へ** をクリックします (図 2.3)。

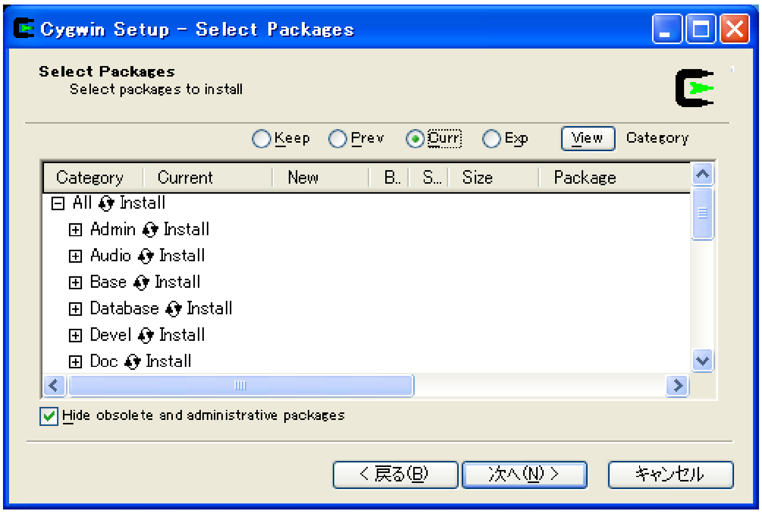


図 2.3　インストールするパッケージの選択

(7) インストール

インストールが開始されます。時間がかかりますので、進度が 100% になるまでしばらくお待ちください。

(8) アイコンの作成

「Create icon on Desktop」を選択して **完了** をクリックします。

「プログラム互換性アシスタント」が表示される場合は、「このプログラムは正しくインストールされました」をクリックします。

(9) ホームディレクトリの初期化

デスクトップ上の「Cygwin Terminal」アイコンをダブルクリックして Cygwin を起動します。最初の起動時にホームディレクトリが初期化されます。

(10) gmake へのリンクの作成

gmake で make が起動するように、Cygwin 上で次のようにシンボリックリンクを作成します。

```
$ cd /usr/bin ↩
$ ln -s make gmake ↩
```

(11) Perl へのリンクの作成

Perl がパス名 `/usr/local/bin/perl` で起動するように、Cygwin 上で次のようにシンボリックリンクを作成します。

```
$ cd /usr/local/bin ↩
$ ln -s /usr/bin/perl ↩
```


(12) 日本語 EUC の設定

もし開発環境を実際に使うアカウントが今までの管理者アカウントと異なる場合は、exit コマンドで Cygwin のウィンドウを閉じ、実際に使うアカウントで Windows にログオンし直してください。デスクトップ上の「Cygwin Terminal」アイコンをダブルクリックして Cygwin を起動します。最初の起動時にホームディレクトリが初期化されます。

ホームディレクトリ上の .bashrc に「export LANG=ja_JP.eucJP」という行を追加します。
例えば次のように操作します。

```
$ cd ↩
$ echo export LANG=ja_JP.eucJP >> .bashrc ↩
```

また、Cygwin のウィンドウ上でマウスの右ボタンをクリックしてメニューを開き、「Options」を選択します。「Text」を選択し、「Locale」を「ja_JP」に、「Character set」を「eucJP」に設定して、[OK] をクリックします。

exit コマンドで Cygwin のウィンドウを閉じます。

2.2 Eclipse のインストール

統合開発環境の Eclipse をインストールします。
詳細については μTeaboard/ARM7-AT91 の CD-ROM 内の『Eclipse インストール方法説明書』をあわせてご参照ください。

(1) Java 実行環境のインストール

Eclipse の実行には Java 実行環境が必要です。まだインストールされていない場合、またはバージョンが古い場合は、http://www.java.com/ からダウンロードしてインストールしてください。

† 64 ビット Windows の場合、32 ビット用の Java 実行環境が必要です。32 ビット用の Java 実行環境をダウンロードしてインストールしてください。

(2) Eclipse システムパッケージの展開

μTeaboard/ARM7-AT91 の CD-ROM 内の共通ソフトウェアのフォルダ (common¥soft) にある以下のファイルをいったんデスクトップなどに保存した上で、Windows のエクスプローラでマウスの右ボタンをクリックして「すべて展開 (A)...」を実行してください。展開先は C ドライブの直下 (C:¥) を指定してください。

Eclipse+CDT	eclipse-cpp-indigo-SR2-incubation-win32.zip

† ファイル名の中のバージョン番号は上記と異なる場合があります。

† 展開先として C:¥ を指定すると、実際には C:¥eclipse というフォルダが作成され、その下に展開されます。

μTeaboard/ARM7-AT91 の CD-ROM 内の日本語版ソフトウェアのフォルダ (jp¥soft) にある以下のファイルをいったんデスクトップなどに保存した上で、Windows のエクスプローラでマウスの右ボタンをクリックして「すべて展開 (A)...」を実行してください。展開先は「C:¥eclipse」を指定してください。

日本語化プラグイン	pleiades_1.3.4.zip

† ファイル名の中のバージョン番号は上記と異なる場合があります。

「C:¥eclipse¥eclipse.ini」をテキストエディタで開き、最後に次の 1 行を追加します。

```
-javaagent:plugins/jp.sourceforge.mergedoc.pleiades/pleiades.jar
```

† 改行コードが LF のため、メモ帳以外のテキストエディタをご使用ください。

(3) ショートカットの作成

「C:¥eclipse¥eclipse.exe」のショートカットをデスクトップ上に作成します。

2.3 T-Kernel 開発環境のインストール

Eclipse に T-Kernel 開発用プラグインを追加インストールします。

詳細については μTeaboard/ARM7-AT91 の CD-ROM 内の『Eclipse 用 T-Kernel 開発環境インストール方法説明書』と『GNU 開発環境 (Eclipse 版) 説明書』をあわせてご参照ください。

(1) T-Kernel 開発用 Eclipse プラグインの展開

μTeaboard/ARM7-AT91 の CD-ROM 内の日本語版ソフトウェアのフォルダ (jp¥soft) にある以下の 2 つのファイルをいったんデスクトップなどに保存した上で、Windows のエクスプローラでマウスの右ボタンをクリックして「すべて展開 (A)...」を実行してください。展開先は「C:¥eclipse」を指定してください。

プラットホーム共通部分	com.t_engine4u.te.1.2.0.zip
各機種対応部分	com.t_engine4u.tl.tbat91.1.1.0.zip

† zip アーカイブの展開は、Windows 標準の zip 展開機能で展開してください。他のソフトを使って展開すると、ソフトによっては正しく展開できない場合があります。

† ファイル名の中のバージョン番号は上記と異なる場合があります。

(2) Eclipse の起動

Eclipse のショートカットをダブルクリックして Eclipse を起動します。図 2.4 のような起動画面が表示され、それからワークスペースの選択ダイアログが表示されます。



図 2.4　Eclipse の起動画面

【トラブル シューティング】「Java Runtime Environment がない」と表示される場合は、Java 実行環境のインストールをご確認ください。

(3) ワークスペースの選択

起動時にワークスペースの選択を聞かれます。ここでは T-Kernel ベースのプログラムを作成するための標準的なワークスペースである「`C:¥te¥kappl`」としてください。

† ワークスペースとは、プロジェクトを保管するフォルダのことです。通常、T-Kernel ベースのプログラムを作成する場合は「`C:¥te¥kappl`」をワークスペースとするのが標準的ですが、これとは違う場所にワークスペースを作成しても構いません。

(4) ワークベンチにジャンプ

「ようこそ」ビューが表示されますので、ワークベンチをクリックします (図 2.5)。

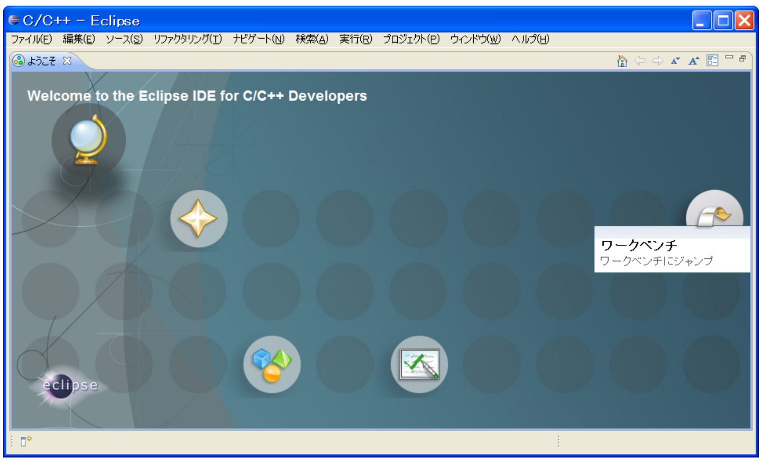


図 2.5　Eclipse の「ようこそ」ビュー

(5) T-Engine 開発パースペクティブを開く

ツールバーの「ウィンドウ」→「パースペクティブを開く」→「その他」 を選択します。「パースペクティブを開く」の中の「T-Engine 開発」を選択して OK をクリックします。(図 2.6)。

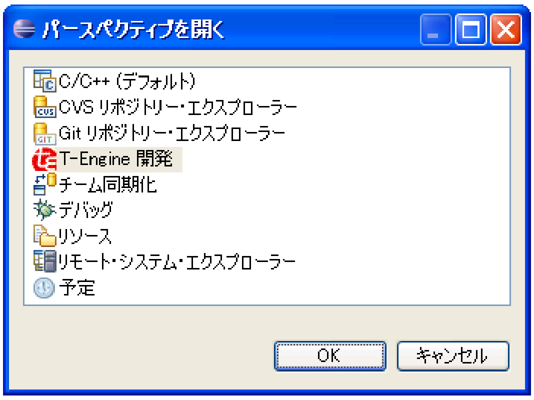


図 2.6 T-Engine 開発パースペクティブの選択

(6) T-Engine 開発環境の設定

ツールバーの「ウィンドウ」→「設定」 で設定ダイアログを開きます。

設定ダイアログの左側の中から「T-Engine 開発環境」を選択して内容を確認します。

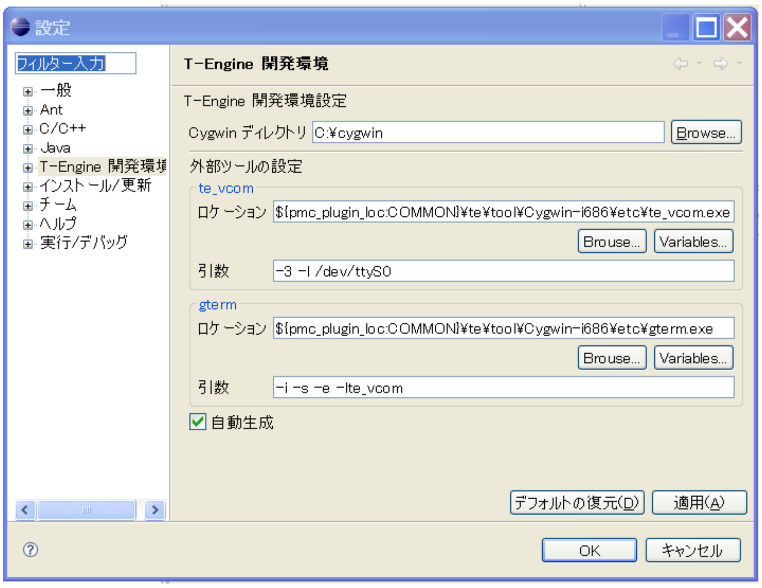


図 2.7 T-Engine 開発環境の設定

Cygwin ディレクトリの欄は、Cygwin をインストールしたディレクトリ (`C:¥cygwin` など) を指定します。

`te_vcom` の引数の欄は、COM ポート番号として COMn を使う場合、`te_vcom` の引数として「`-3 -l /dev/ttyS`$(n-1)$」を指定します。

† `-3` は 通信速度 38400bps、`-l` はマイナス・エルで、ラインデバイス名を指定します。COM ポート番号から 1 を引いた値がラインデバイス名になります。例えば COM1 であれば「`-3 -l /dev/ttyS0`」、COM2 であれば「`-3 -l /dev/ttyS1`」を設定してください。

(7) ワークスペースの設定

設定ダイアログの左側の中から「一般」の「ワークスペース」を選択して、次の設定を行います (図 2.8)。

- 「自動的にビルド」はチェックを入れた状態にします。
- 「テキスト・ファイル・エンコード」の「その他」をチェックして、「`EUC-JP`」を選択します。選択肢にない場合は直接「`EUC-JP`」とキー入力します。
- 「新規テキスト・ファイルの行区切り文字」の「その他」をチェックして、「`Unix`」を選択します。

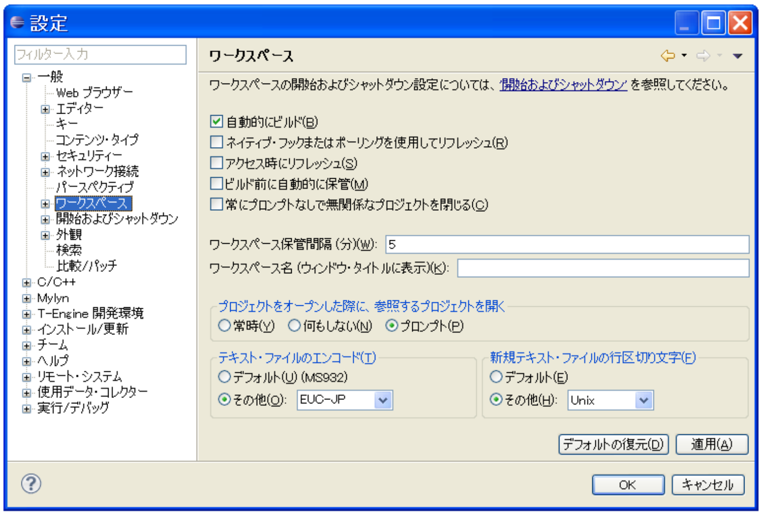


図 2.8 ワークスペースの設定

[OK] をクリックして設定ダイアログを閉じます。

なお環境設定はワークスペースごとに行う必要があります。他のワークスペースに切り替えた際には、同様の設定を行ってください。

(8) ターゲット側との接続の確認

ターゲットと通信を行うため、中継プログラム (`te_vcom`) とターミナルエミュレータ (`gterm`) を起動します。

ツールバーの「実行」→「外部ツール」→「外部ツールの構成…」を選択します。外部ツール構成のダイアログが表示されたら、左側の「プログラム」の左の「+」をクリックすると `te_vcom` と `gterm` の設定が表示されます。設定内容は通常変更する必要はありません。

まずは `te_vcom` を選択し、[**実行**] をクリックします (図 2.9)。

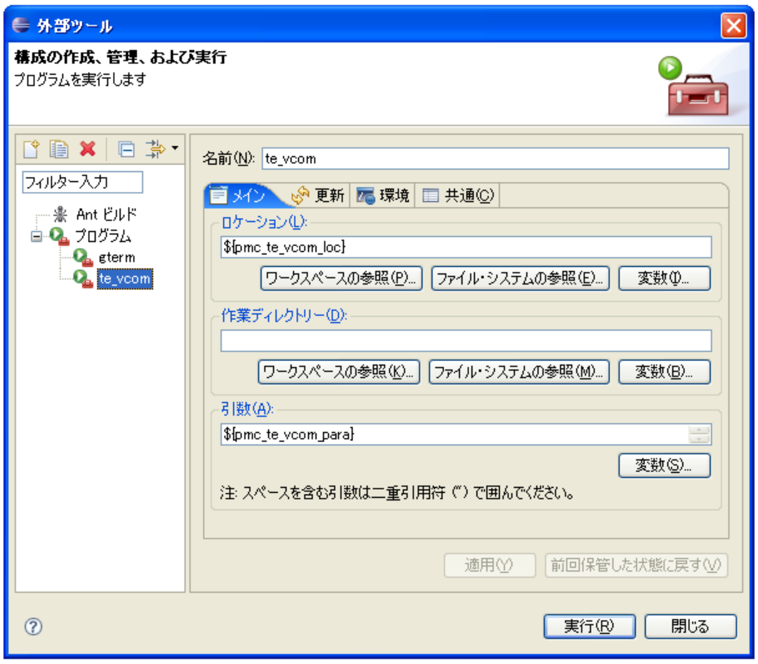


図 2.9 te_vcom の起動

続いて同様の手順で外部ツール構成のダイアログから gterm を起動してください。コンソールに gterm の起動メッセージが表示されます。

ターゲット側の μTeaboard が起動した状態で、gterm のコンソール上で ←(Enter) キーを入力すると、ターゲット側から返されるプロンプト「[IMS]%」が表示されます (図 2.10)。

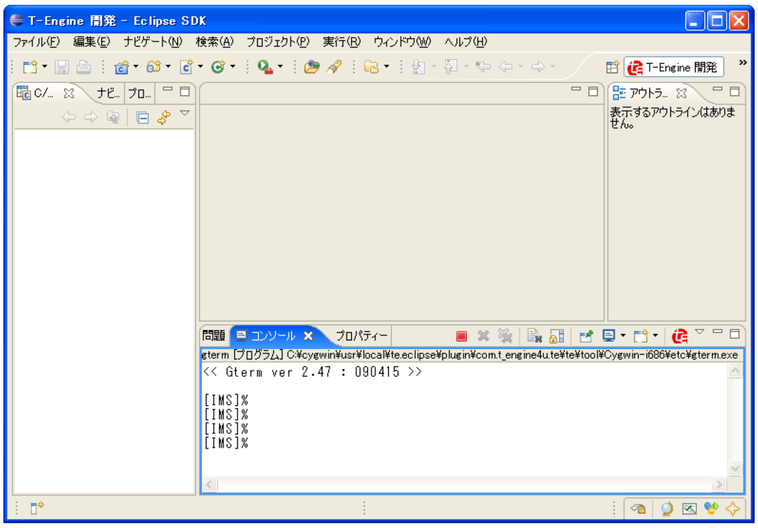


図 2.10 gterm 上に表示されたターゲット側プロンプト

これらの外部ツール te_vcom と gterm は、一度起動すると次回からはツールバーの外部ツールのプルダウンリストにショートカットが表示されますので、ショートカットを使って起動できます。

3 開発手順

3.1 ターゲット側と開発環境側の起動

開発を始めるには、まずターゲット側と開発環境側をそれぞれ次の手順で起動します。

(1) ターゲット側 (μTeaboard) の起動

μTeaboard のディップスイッチ (DSW1) は 4 つとも OFF にします。
μTeaboard に 5V の AC アダプタを接続して電源を供給します。
μTeaboard の LED5 (緑) および LED1 (赤) が点灯します。

(2) Eclipse の起動

Eclipse を起動します。

最初にワークスペースの選択を聞かれますので、T-Kernel ベースの場合は「`C:¥te¥kappl`」とします。

(3) `te_vcom` と `gterm` の起動

μTeaboard と通信を行うには、メニューバーの外部ツールの▼をクリックして、中継プログラム (`te_vcom`) と端末エミュレータ (`gterm`) を起動しておく必要があります。

† Eclipse 上の開発では、通常はコンソールからコマンドを手で入力しなくても、Eclipse のダイアログからプログラムの転送や実行ができるようになっています。しかしこの場合でも Eclipse と μTeaboard の間の通信はコンソールを使いますので、通信が確立されていないとプログラムの転送や実行もできません。
したがって `te_vcom` や `gterm` を必ず起動しておく必要があります。とくに Eclipse を起動した直後や、ワークスペースを切り替えた直後は、`te_vcom` や `gterm` は起動されていませんので、毎回起動する必要があります。

Eclipse のコンソールウィンドウの `gterm` で、←↩(Enter) キーを何回か押して、IMS のプロンプト「`[IMS]%`」が表示されることを確認します。

† 複数の端末ソフトを同時に μTeaboard に接続しないでください。具体的には、Eclipse 上で `te_vcom` や `gterm` を起動する際は、ハイパーターミナルや TeraTerm などの他の端末ソフトは接続しない状態にしてください。また `te_vcom` や `gterm` を 2 個以上重ねて起動しないでください。

【トラブル シューティング】IMS のプロンプトが出ていない場合 (T-Monitor のプロンプト「`TM>`」が出ているなど) は、μTeaboard のリセットスイッチ (ボード中央付近の SW5) を押してターゲット側を再起動してください。

3.2 コンソールの利用

μTeaboard は開発に便利な各種ツール類が付属しています。プログラムの転送や実行については、`gterm` のコンソールからコマンドを手で入力しなくても、Eclipse のダイアログからプログラムの転送や実行ができるようになっています。しかしそれ以外のさまざまな用途のために、必要に応じて `gterm` のコンソールからこれらのツール類をご利用ください。

IMS (Initial Monitor System)

T-Kernel ベースのモニタです。プロンプトは標準では「[IMS]%」です。詳しくは『μTeaboard 取扱説明書』の「5 章 IMS」をご覧下さい。

T-Monitor

最も基本的なモニタです。ハードウェアレベルの操作などに威力を発揮します。プロンプトは「TM>」です。

T-Monitor のコマンド一覧は『T-Monitor 仕様書』の「3.3. コマンド一覧」をご覧下さい。さらに μTeaboard で追加された機能もあります。追加されたコマンドは『実装仕様書』の「2. T-Monitor 実装仕様」をご覧下さい。

以下に IMS と T-Monitor の簡単な実習例を示します。

(1) IMS のプロンプト確認

Eclipse のコンソールウィンドウの gterm 上で ←(Enter) キーを何回か押して、IMS のプロンプト「[IMS]%」が表示されることを確認します。

```
[IMS]%
```

(2) IMS コマンド一覧

IMS の "?" コマンドを入力して、IMS のコマンド一覧を表示させます。

```
[IMS]% ?←
                        < IMS Commands>
ref_tsk   ref_sem   ref_flg   ref_mbx …(後略)…
```

(3) IMS コマンド詳細

IMS の "?" コマンドを使って "ref_tsk" コマンドの詳細ヘルプを表示させます。

```
[IMS]% ?  ref_tsk←
ref_tsk       : Display All Task Status
ref_tsk tskid : Display Specified Task Status
```

(4) タスク一覧

IMS の "ref_tsk" コマンドを使ってタスクの一覧を表示させます。

```
[IMS]% ref_tsk←
TSK STATE (MAX:150)
TID PRI:BPR SLT WUP SUS STS(*:NODISWAI) ST+UT(x10) RID EXINF/NAME
  1 138:138   0   0   0 RUN             106+0        1 ffffffff
  2   6:  6   0   0   0 WAI-RMBF *[  1]   0+0        1 676f4c53 SLog
… (後略) …
```

(5) T-Monitor への移行

IMS の「#」コマンドを使って T-Monitor へ移行します。

```
[IMS]% #↩
TM>
```

(6) T-Monitor コマンド一覧

T-Monitor の “?” コマンドを使って T-Monitor のコマンド一覧を表示させます。

```
TM> ?↩
--- Command List :   "? command" for details ---
DumpByte/Half/Word(D/DB/DH/DW)  … (後略) …
```

(7) T-Monitor コマンド詳細

T-Monitor の “?” コマンドを使って “OW” コマンドの詳細ヘルプを表示させます。

```
TM> ?  ow↩
OutputWord(OW) port,data : Output Word to I/O port
```

(8) I/O 書き込み

T-Monitor の “ow” コマンドを使って `0xffff0030` 番地 (消灯) または `0xffff0034` (点灯) に 32 ビットの値を書き込んで、LED1~LED4 (赤) を制御します。

```
TM> ow ffff0030,100000↩ — LED1 を消灯
Port FFFF0030:W <-- 00100000
TM> ow ffff0034,100000↩ — LED1 を点灯
Port FFFF0034:W <-- 00100000
```

LED	PIO	値
LED1	PB20	0x100000
LED2	PB21	0x200000
LED3	PB23	0x800000
LED4	PB24	0x1000000

(9) T-Monitor からの復帰

T-Monitor の “G” (Go) コマンドを使って IMS へ復帰します。

```
TM> g↩
[IMS]%
```

3.3 T-Kernel ベースのプログラム

T-Kernel 上のプログラムは、大きく分けると「T-Kernel ベースのプログラム」と「プロセスベースのプログラム」の二つに分類されます。ただし μTeaboard には MMU (メモリ管理ユニット) がないため、プロセスベースのプログラムはサポートしていません。T-Kernel ベースのプログラムの特徴は次の通りです。

T-Kernel ベースのプログラム

T-Kernel の機能を直接使うプログラムです。

ハードウェア制御や割込みなどを扱うことができます。デバイスドライバなどを作成することができます。

メモリ保護は効きません。

メモリ空間はシステム全体で一つの空間を共有し、その中に複数の T-Kernel ベースのプログラムが配置される形です。

T-Kernel ベースのプログラムは `lodspg` コマンドで起動を行います。この時に main() 関数が呼ばれ、第一引数 `ac` は渡される引数の個数を示します。通常この中でタスクやハンドラの生成などの初期化処理を行います。

また `unlspg` コマンドでプログラムを終了することができます。この時にも main() 関数が呼ばれ、第一引数 `ac` はマイナスの値です。通常この中で、最初に生成したタスクやハンドラの削除などの終了処理を行います。

T-Kernel ベースのプログラムから使える API については、CD-ROM 内の以下のドキュメントをご参照ください。

- **『T-Kernel 仕様書』**
  タスクやセマフォ、割込みハンドラなどのリアルタイム OS としての基本機能 (T-Kernel/OS) と、デバイスドライバ管理などの機能 (T-Kernel/SM) などの仕様書です。
- **『ライブラリ説明書』**
  C 言語標準ライブラリをはじめとするライブラリの説明書です。
- **『デバイスドライバ説明書』**
  デバイスドライバを作成したりデバイスドライバを呼び出す場合に必要となる説明書です。
- **『TCP/IP マネージャ説明書』**
  ネットワーク (TCP/IP) のプログラミングを行う際に必要となる説明書です。

3.4 サンプルプログラム: Hello, world

有名な「Hello, world」のプログラムを、T-Kernel ベースで作成してみましょう。プログラムの初期化時に「Hello, world」と表示し、終了時に「See you again」と表示するようにします。

(1) ワークスペースの指定

T-Kernel ベースのプログラムですので、Eclipse 起動時にワークスペースとして「`C:¥te¥kappl`」を指定してください。

既に別のワークスペースで Eclipse が起動している場合は、メニューバーの「ファイル」→「ワークスペースの切り替え」でワークスペースを「`C:¥te¥kappl`」に切り替えてください。

† ワークスペースを切り替えると Eclipse がいったん終了して te_vcom と gterm も終了しますので、再度 te_vcom と gterm を順番に起動する必要があります。その上で Eclipse のコンソールウィンドウの gterm で、←(Enter) キーを何回か押して、IMS のプロンプト [IMS]% が表示されることを確認しておきます。

(2) プロジェクトの新規作成

テンプレートを使ってプロジェクトを新規作成します。

「C/C++ プロジェクト」ビュー内で右クリックして (またはメニューバーの「ファイル」から)、「新規」→「T-Engine C/C++ プロジェクト」を選択して、新規プロジェクトダイアログで以下のように入力します (図 3.1)。

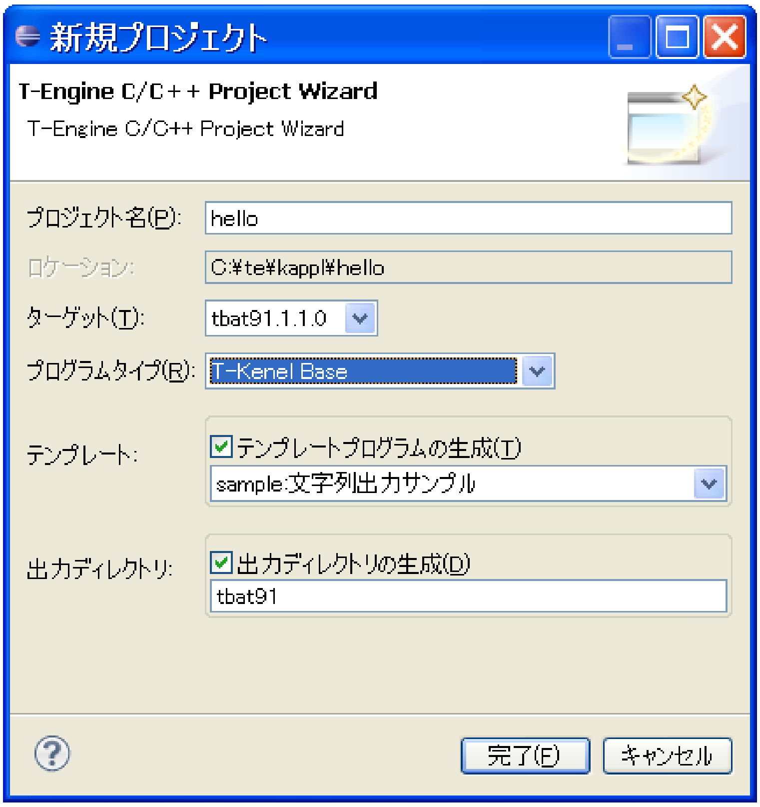


図 3.1　新規プロジェクト作成ダイアログ

- プロジェクト名:
  プロジェクト名は自由につけることができますが、ここでは例えば「hello」とします。
- ターゲット:
  ターゲットの機種 (tbat91) を選択します。
- プログラムタイプ:
  「T-Kernel Base」を指定します。ワークスペースを「C:¥te¥kappl」としていれば、「ワークスペース名による自動決定」のままでも自動的に T-Kernel ベースになります。

- テンプレート:
  チェックを入れて、テンプレートをここでは「sample:文字列出力サンプル」を選択します。
- 出力ディレクトリの生成:
  チェックを入れます。

最後に **完了** をクリックするとプロジェクトが自動生成されます。

(3) ソースの作成

「C/C++ プロジェクト」ビュー内に「hello」プロジェクトが生成されますので、ダブルクリックして開くとソースプログラムなどが参照できます。

ソースを次のように修正して、「Hello, world」を作成しましょう。

- hello/src/main.c :
  たとえば以下のように修正してください。

```
/* Hello, world (T-Kernel ベース) */
#include <basic.h>       /* 基本共通ヘッダ */
#include <tk/tkernel.h>  /* T-Kernel ヘッダ */
#include <stdio.h>       /* printf() など */

ER main( INT ac, UB *av[] )
{
    if (ac >= 0) {
        printf("Hello, world¥n");
    } else {
        printf("See you again¥n");
    }
    return E_OK;
}
```

  修正したら、メニューバーの「ファイル」→「保管」で修正を保存します。

- hello/src/Makefile :
  作成対象は「TARGET = hello」として、作成対象 (メイクしてできる実行ファイル名) を hello に変更します。
  ifeq ($(MACHINE), tbat91) で始まる部分は次のように修正します。これはプログラムを μTeaboard の ROM 上ではなく RAM 上に転送する指定です。

```
ifeq ($(MACHINE), tbat91)
  TK_ONLY = 1
  START_ADR = -Wl,-Ttext,0x20100000,-Tdata,0x20110000
endif
```

  修正したら、メニューバーの「ファイル」→「保管」で修正を保存します。

(4) メイク

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「hello/tbat91/Makefile」を選択した上で、メニューバーの「プロジェクト」→「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「`hello/tbat91/`」の下に「`hello`」という名前で、絶対アドレス形式の実行ファイルが生成されます (▷で表示されます)。このほか転送用ファイルとして `hello.mot` も作成されます。

【トラブル シューティング】「ビルダー起動中のエラー」が発生する場合は、「`hello/tbat91/Makefile`」を選択しているか、いまいちどご確認ください。

【トラブル シューティング】「makedeps: Command not found」と出る場合は、`/usr/local/bin/perl` に Perl が正しくインストールされていない可能性があります。Cygwin のインストールおよびインストール後の設定をご確認ください。

(5) 実行ファイルの転送とプログラム起動

転送する実行ファイルとして「C/C++ プロジェクト」ビュー内の「`hello/tbat91/hello`」を選択した上で、右クリックメニューの「実行」→「実行の構成」を選択します。

実行構成ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「`hello`」の転送やリモート実行方法などの設定が作成されます。

「メイン」タブの「転送ファイル」欄は空欄になっていますが、転送用ファイルとして
「`tbat91¥hello.mot`」を指定してください (図 3.2)。

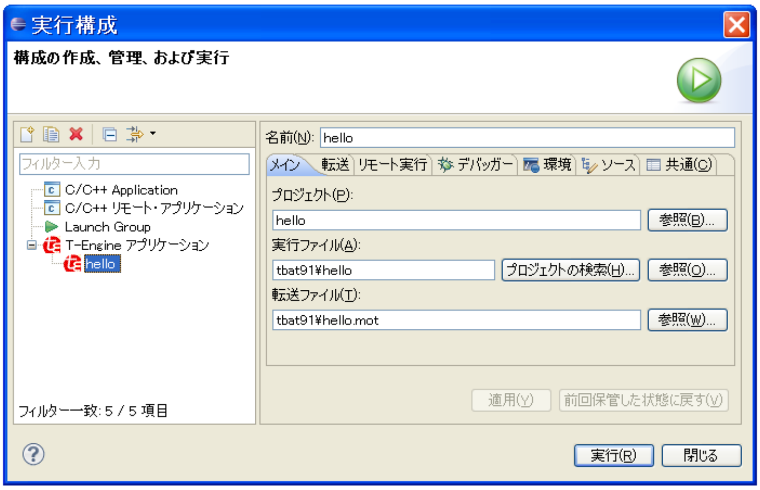


図 3.2 「構成および実行」ダイアログ

実行 をクリックすると、自動的に転送 (`.load` コマンド) とプログラム起動 (`lodspg` コマンド) が行われ、「Hello, world」が表示されます (図 3.3)。

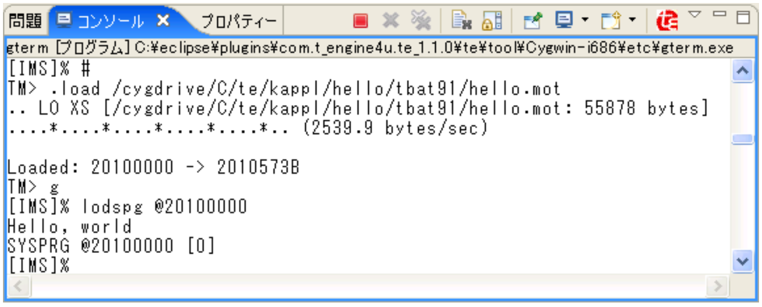


図 3.3　「Hello, world」の実行

【トラブル シューティング】「No load address」が発生する場合は、転送用ファイルとして
「`tbat91¥hello.mot`」を指定しているか、いまいちどご確認ください。

(6) プログラム終了

`unlspg` コマンドでプログラムを終了します。「See you again」と表示されます。

```
[IMS]% unlspg @20100000↩
See you again
[IMS]%
```

pagebreak

4　実習用サンプルプログラム

μTeaboard/ARM7-AT91 の CD-ROM 内のチュートリアルのページ (`jp¥tutorial`) には、以下の実習用サンプルプログラムのソースが付属しています。

- LED 制御サンプル
- ネットワーク送受信サンプル

4.1　LED 制御サンプル

LED の点滅パターンを制御する T-Kernel ベースのプログラムを作成します。

ブレークポイントをかけて止めながら少しずつ動かすデバッグのサンプルとしても使えます。

(1) ワークスペースの指定

T-Kernel ベースのプログラムですので、Eclipse 起動時にワークスペースとして「`C:¥te¥kappl`」を指定してください。

既に別のワークスペースで Eclipse が起動している場合は、メニューバーの「ファイル」→「ワークスペースの切り替え」でワークスペースを「`C:¥te¥kappl`」に切り替えてください。

† ワークスペースを切り替えると Eclipse がいったん終了して te_vcom と gterm も終了しますので、再度 te_vcom と gterm を順番に起動する必要があります。その上で Eclipse のコンソールウィンドウの gterm で、↩(Enter) キーを何回か押して、IMS のプロンプト [IMS]% が表示されることを確認しておきます。

(2) プロジェクトの新規作成

プロジェクトを新規作成します。今回はテンプレートは使用しません。

「C/C++ プロジェクト」ビュー内で右クリックして (またはメニューバーの「ファイル」から)、「新規」→「T-Engine C/C++ プロジェクト」を選択して、新規プロジェクトダイアログで以下のように入力します。

- プロジェクト名:
  プロジェクト名は自由につけることができますが、ここでは例えば「led」とします。
- ターゲット:
  ターゲットの機種 (tbat91) を選択します。
- プログラムタイプ:
  「T-Kernel Base」を指定します。ワークスペースを「C:¥te¥kappl」としていれば、「ワークスペース名による自動決定」のままでも自動的に T-Kernel ベースになります。
- テンプレート:
  チェックを入れません。
- 出力ディレクトリの生成:
  チェックを入れます。

最後に **完了** をクリックするとプロジェクトが自動生成されます。

(3) ソースの作成

「C/C++ プロジェクト」ビュー内に生成されたプロジェクト「led」を選択して、次のようにソースをインポートします。

- メニューバーの「ファイル」→「インポート」を選択して、「インポート」ダイアログを開きます。
- 「一般」の左の + をクリックして開き、「アーカイブファイル」を選択して、**次へ** をクリックします。
- ソースアーカイブファイル欄の右の **参照** をクリックして「アーカイブファイルからインポート」ダイアログを開き、μTeaboard/ARM7-AT91 の CD-ROM 内の実習用プログラム「led.zip」を選択して **開く** をクリックします。
- **完了** をクリックします。

「led/src」内にソースファイル「main.c」とメイクファイル「Makefile」がインポートされます。

(4) メイク

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「led/tbat91/Makefile」を選択した上で、メニューバーの「プロジェクト」→「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「led/tbat91/」の下に「led」という名前で、絶対アドレス形式の実行ファイルが生成されます (▷で表示されます)。このほか転送用ファイルとして led.mot も作成されます。

(5) 実行ファイルの転送とプログラム起動

転送する実行ファイルとして「C/C++ プロジェクト」ビュー内の「led/tbat91/led」を選択した上で、右クリックメニューの「実行」→「実行の構成」を選択します。

実行構成ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「led」の転送やリモート実行方法などの設定が作成されます。

「メイン」タブの「転送ファイル」欄は空欄になっていますが、転送用ファイルとして「tbat91¥led.mot」を指定してください。

実行 をクリックすると、自動的に転送 (.load コマンド) とプログラム起動 (lodspg コマンド) が行われ、LED が点滅します。

(6) デバッグ用のメイク

デバッグを行うためには、デバッグ情報つきの実行ファイルをメイクする必要があります。「tbat91.debug」という名前のフォルダ上でメイクすれば、デバッグオプションがついて、デバッグ情報つきでメイクされます。

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「led/tbat91.debug/Makefile」を選択した上で、ツールバーの「プロジェクト」→ 「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「led/tbat91.debug/」の下に「led」という名前で、デバッグ情報つきの実行ファイルが生成されます (虫の図形で表示されます)。このほか転送用ファイルとして led.mot も作成されます。

(7) 他のプロジェクトを閉じる

今回のデバッグ対象である led 以外のプロジェクトが開いている場合は、右クリックメニューの「プロジェクトを閉じる」で、すべて閉じてください。

(8) デバッグの開始

デバッグするデバッグ情報つき実行ファイルとして「C/C++ プロジェクト」ビュー内の「`led/tbat91.debug/led`」を選択した上で、右クリックメニューの「デバッグ」→ 「デバッグの構成」を選択します。

デバッグ構成ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「`led`」の転送やデバッグ方法などの設定が自動設定されます。

† (5) で作成した実行のための設定も残っていますが、今回はデバッグを行うので、必ず新規作成してください。

「メイン」タブの「転送ファイル」欄は空欄になっていますが、転送用ファイルとして「`tbat91.debug¥led.mot`」を指定してください。

デバッグ をクリックすると、転送とデバッグが開始され、デバッグパースペクティブに切り替わります。切り替えの確認ダイアログが出る場合は **はい** で切り替えてください。

(9) ブレークポイントの設定

現在 main() 関数の先頭で停止した状態です。

ここで task1 の中の「`tk_dly_tsk( 50 );`」という行にブレークポイントをかけましょう。この行の左端でマウスを右クリックしてメニューを出し、「ブレークポイントの切り替え」でブレークポイントを設定します (図 4.1)。

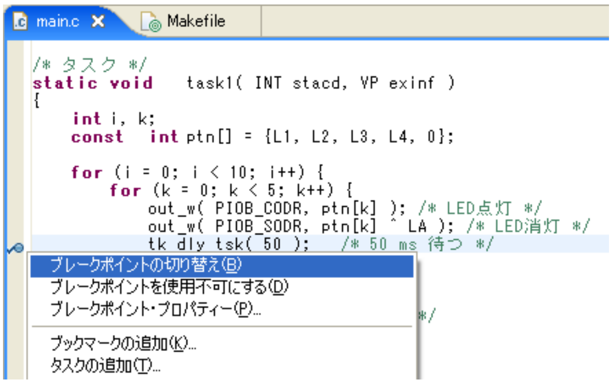


図 4.1 ブレークポイントの設定

(10) 実行再開

デバッグビュー内の再開ボタン (緑の右の三角) をクリックして実行再開すると、ブレークポイントで停止するまで実行します (図 4.2)。



図 4.2 実行再開

再開を繰り返すと、μTeaboard 上では 1 ステップずつ LED の点灯パターンが動いていくのが分かります。

(11) ステップ実行

また、デバッグビュー内の「ステップオーバー」ボタン (飛び越し矢印) をクリックすると、1 行のみ実行して、次の行で停止します。

† 今回は「ステップオーバー」を使ってください。「ステップイン」は、関数の中まで追跡しようとします。

(12) デバッグの終了

デバッグビュー内の終了ボタン (赤い四角) をクリックします。

さらにデバッグビュー内の「終了したすべての起動を除去」のボタン (灰色の 2 重×) をクリックします。

最後に Eclipse のウィンドウの右上にある「》」を選択して「T-Engine 開発」を選択すれば、元の T-Engine 開発パースペクティブに戻ります。

4.2 ネットワーク送受信サンプル

ネットワーク (TCP/IP) で送受信を行う T-Kernel ベースのプログラムを作成します。ここでは μTeaboard 上に HTTP サーバのサブセットを作成し、パソコン上のブラウザとの間でネットワーク経由で送受信を行います。

(1) ワークスペースの指定

T-Kernel ベースのプログラムですので、Eclipse 起動時にワークスペースとして「`C:¥te¥kappl`」を指定してください。

既に別のワークスペースで Eclipse が起動している場合は、メニューバーの「ファイル」→「ワークスペースの切り替え」でワークスペースを「`C:¥te¥kappl`」に切り替えてください。

† ワークスペースを切り替えると Eclipse がいったん終了して `te_vcom` と `gterm` も終了しますので、再度 `te_vcom` と `gterm` を順番に起動する必要があります。その上で Eclipse のコンソールウィンドウの `gterm` で、↩(Enter) キーを何回か押して、IMS のプロンプト `[IMS]%` が表示されることを確認しておきます。

(2) プロジェクトの新規作成

プロジェクトを新規作成します。今回はテンプレートは使用しません。

「C/C++ プロジェクト」ビュー内で右クリックして (またはメニューバーの「ファイル」から)、「新規」→「T-Engine C/C++ プロジェクト」を選択して、新規プロジェクトダイアログで以下のように入力します。

- プロジェクト名:
  プロジェクト名は自由につけることができますが、ここでは例えば「net」とします。
- ターゲット:
  ターゲットの機種 (tbat91) を選択します。
- プログラムタイプ:
  「T-Kernel Base」を指定します。ワークスペースを「C:¥te¥kappl」としていれば、「ワークスペース名による自動決定」のままでも自動的に T-Kernel ベースになります。
- テンプレート:
  チェックを入れません。
- 出力ディレクトリの生成:
  チェックを入れます。

最後に **完了** をクリックするとプロジェクトが自動生成されます。

(3) ソースの作成

「C/C++ プロジェクト」ビュー内に生成されたプロジェクト「net」を選択して、次のようにソースをインポートします。

- メニューバーの「ファイル」→「インポート」を選択して、「インポート」ダイアログを開きます。
- 「一般」の左の + をクリックして開き、「アーカイブファイル」を選択して、**次へ**をクリックします。
- ソースアーカイブファイル欄の右の **参照** をクリックして「アーカイブファイルからインポート」ダイアログを開き、μTeaboard/ARM7-AT91 の CD-ROM 内の実習用プログラム「net.zip」を選択して **開く** をクリックします。
- **完了** をクリックします。

「net/src」内にソースファイル「main.c」とメイクファイル「Makefile」がインポートされます。

(4) ネットワークの設定

「net/src/main.c」を開き、その中の

```
# define MY_IP "192.168.0.100" // 自 IP アドレス
```

の部分を、μTeaboard 側の IP アドレスとして使用できる値に修正します。

修正したら、メニューバーの「ファイル」→「保管」で保存します。

† ご使用のネットワーク (LAN) 環境でローカルに使用可能な固定 IP アドレスを設定してください。IPv4 のみに対応しております。
LAN に接続したくない場合は、パソコンと μTeaboard のみを直接 LAN ケーブルで接続し、パソコン側と μTeaboard 側に異なる固定 IP アドレスを割り当てても構いません。

(5) メイク

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「net/tbat91/Makefile」を選択した上で、メニューバーの「プロジェクト」→「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「net/tbat91/」の下に「net」という名前で、絶対アドレス形式の実行ファイルが生成されます (▷で表示されます)。このほか転送用ファイルとして net.mot も作成されます。

(6) 実行ファイルの転送とプログラム起動

転送する実行ファイルとして「C/C++ プロジェクト」ビュー内の「net/tbat91/net」を選択した上で、右クリックメニューの「実行」→「実行の構成」を選択します。

実行構成ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「net」の転送やリモート実行方法などの設定が作成されます。

「メイン」タブの「転送ファイル」欄は空欄になっていますが、転送用ファイルとして「tbat91¥net.mot」を指定してください。

実行 をクリックすると、自動的に転送 (.load コマンド) とプログラム起動 (lodspg コマンド) が行われます。

(7) ブラウザからの閲覧

同じネットワーク (LAN) 上のパソコンのブラウザから、μTeaboard の IP アドレスを指定して、ルートにアクセスします。例えば μTeaboard の IP アドレスが 192.168.0.100 であれば、ブラウザから次のように URL を入力して下さい:

「http://192.168.0.100/」

μTeaboard 側からの応答として、ブラウザ上に「Hello, world」と閲覧回数が表示されます。

ブラウザの再読み込みを行うと、閲覧回数が増加します。

索 引

はじめてみよう μTeaboard/ARM7-AT91
Version 1.01

パーソナルメディア株式会社
Web: `http://www.t-engine4u.com/`
E-Mail: `te-sales@personal-media.co.jp`